2011 International Conference on Management and Artificial Intelligence
IPEDR vol.6 (2011) © (2011) IACSIT Press, Bali, Indonesia

Simple Bacteria Cooperative Optimization with Rank-based Perturbation

Sung Hoon Jung
Department of Information and Communications Engineering
Hansung University
Seoul 136-792, Korea
shjung@hansung.ac.kr

***Abstract*—Simple bacteria cooperative optimization (sBCO) algorithm has shown relatively good performances, but their performances were limited by step-by-step movement of individuals. In order to solve this limitation, this paper introduces sBCO algorithm with rank-based perturbation. In existing sBCO algorithms, individuals moved the playground without any perturbations. This caused the sBCO to fall into local optimum areas and resulted in poor performances of the sBCO. Rank-based perturbation through the variable speed and mutation of individuals can decrease the probability of falling into local optimum areas. This prevents the individuals from long staying in the local optimum areas. We tested the proposed algorithm with four function optimization problems. It was found from extensive experiments that the proposed perturbation enabled the sBCO not to fall into local optimum areas and to fast approach to the global optimum areas.**

Keywords-optimization; bacteria cooperative optimization; rank-based perturbation

I. INTRODUCTION

Recently bio-inspired algorithms such as ant colony optimization, artificial immune system, and particle swarm optimization have been widely used for engineering applications [1-3]. As another bio-inspired optimization algorithm, we introduced the simple bacteria cooperative optimization (sBCO) based on the modeling of chemotaxis of E. coli [4]. The sBCO has shown relatively good performances, but sometimes bad performances especially in cases of small individuals because the individuals (in other words, artificial E. colis, abbreviated into AEs) easily fell into local optimum areas. This was because AEs simply moved one step at a time without any perturbation. As a perturbation method, we applied a rank replacement strategy that replaced a specific percent of bad AEs with newly generated AEs [5]. Although its performance was somewhat improved through getting out of local optimum areas by newly generated AEs, it was not an efficient method because newly generated AEs were distributed within whole ranges of playground.

In this paper, we introduce another revised sBCO by employing the variable speed and mutation of AEs as a perturbation method. Unlike existing sBCO, AEs in this method move variable steps corresponding to their ranks at a time. From this, highly ranked AEs move slowly in order to steady approach to global optimum areas while lowly ranked AEs move fast in order to find new good areas. Also, bad individuals are mutated within some ranges that are proportion to their ranks. That is, the lower the ranks of individuals are, the larger the mutation distances are. This enables the bad individuals to have more chances to move to good places. Since good individuals with good ranks do not need such mutation, a specific percent of bad individuals are mutated. We tested the proposed algorithm with four function optimization problems and observed the experimental results. It was observed that our proposed algorithm helps the sBCO not to fall into local optimum areas and to get out of the local optimum areas and fast approach to the global optimum areas. Finally, the sBCO with rank-based perturbation showed better performances than the existing sBCO.

II. SIMPLE BACTERIA COOPERATIVE OPTIMIZATION

We introduced simple bacteria cooperative optimization (sBCO) based on the modeling of behavior patterns for foraging of E. coli [4]. E. coli, a kind of bacteria, has been evolved for few millenniums for foraging. From this we thought that their behavior patterns for foraging, in other words chemotaxis, might be optimized to their environments. Based on this observation we derived two simple rules for foraging from their real behavior patterns [6]. The two kinds of simple rules, behavior rules and decision rules, are the key operations of the sBCO.

Behavior rules are given: (B1) AEs decide their actions for run or tumble every B_n runs; (B2) if their run counts become to $B_m (> B_n)$, then AEs must do tumble. First behavior rule B1 is concerned with the distribution of attractant chemical molecules for foraging. That is, if the density of attractant chemical molecules is gradually increased, then the large B_n is better than the small one, but it also increases AEs to fall into local optimum areas. Otherwise, the small B_n is better than the large one in order to fast find new good direction. Second behavior rule B2 has two effects: first is to find better direction even if the current direction is good; and second is to prevent AEs from straight going to optimum areas. If the optimum areas have global optima, then it is not a problem; otherwise it causes the sBCO fall into local optimum areas. If the local optimum areas are wide enough not to get out of the areas, then individuals within the areas stay there for a long time. In multimodal functions having a lot of local optimum areas, this caused a fatal problem. However, only behavior rule B2 is not enough to prevent AEs from falling local optimum areas because the new direction getting out of the local optimum areas by the rule B2 may soon change to the

direction of local optimum areas. As a result, another perturbation method to alter the individual positions not to stay in local optimum areas must be devised.

Decision rules are given: (D1) AEs calculate the current density of attractant chemical molecules using the average values of those on D_n steps and the previous density using the average values of those on $D_m (> D_n)$ steps on the discrete playground; (D2) if the current density of attractant chemical molecules is greater than the previous density, then the AE decides its action to run, otherwise, tumble. First decision rule D1 is related to the resolution of checking the attractant chemical molecules for foraging. That is to say, if the D_n and D_m are small, then AEs sensitively respond the variation of the attractant chemical molecules; otherwise, they will insensitively respond. If optimized functions have global optima whose regions are narrow convex, then the small D_n and D_m may be better than large ones and vice versa.

Algorithm 1) Simple Bacteria Cooperative Optimization with Rank-based Perturbation

```
// t : discrete time
// v: speed of AE corresponding to the rank of AE
// r_c: the run count
// B_n: the minimum number of runs to decide the actions of AEs
// B_m: the maximum number of runs to go straight without tumble
// D_n: the number of steps for measuring current density of attractant molecules
// D_m: the number of steps for measuring previous density of attractant molecules
// ρ_Dn: the current density of attractant chemical molecules
// ρ_Dm: the previous density of attractant chemical molecules
// E(t): AEs at time t
  t = 0
  initialize E(t)
    make AEs at random positions uniformly distributed within operating ranges
    set initial directions of all AEs to random direction among eight directions
    set initial modes of all AEs to run
    set r_c, ρ_Dn, and ρ_Dm of all AEs to zero
    sense and store the amount of attractant chemical molecules at current position
  while (not termination-condition)
  do
    t = t + 1
    rank E(t)
      rank all AEs
      assign the speed v of all AEs according to the ranks
    move E(t)
      move each AE with v steps of playground to their directions
      increase r_c of each AE
    sense E(t)
      sense and store the amount of attractant chemical molecules at current position
      calculate ρ_Dn and ρ_Dm                          ▷ decision rule (D1)
    decide E(t)
      if (r_c mod B_n) = 0 then                        ▷ behavior rule (B1)
        if ρ_Dn > ρ_Dm then                            ▷ decision rule (D2)
          set mode to run
        else
          set mode to tumble
        end if
      end if
      if r_c = B_m then                                ▷ behavior rule (B2)
        set mode to tumble
      end if
      if tumble mode then
        set direction to random direction except for current and opposite directions
        set run mode
        set r_c = 0
      end if
    mutate E(t)
      select a specific percent of bad AEs
      mutate selected AEs within some ranges in proportion to their ranks
  end
```

These two rules are key operations for foraging of AEs. Based on these two rules, we devised a simple bacteria cooperative optimization algorithm in [4].

III. sBCO WITH RANK-BASED PERTURBATION

Although the sBCO algorithm showed relatively good performances, its performance was limited because the AEs moved only one step at a time. This caused the AEs to slow move to the global optimum areas and more fatally made the AEs fall into local optimum areas and not to get out of the areas.

This makes the AEs stay in the local optimum areas for a long time and results in poor performances especially in the cases of small individuals. As a perturbation method, we introduced a rank replacement method that replaced a specific percent of bad AEs with newly generated AEs. This perturbation considerably improved the performances of sBCO, but it was not an efficient method because newly generated AEs are distributed the whole ranges of playground. From this observation, we devised the variable speed and mutation of AEs based on their ranks as another perturbation method in this paper. AEs move variable steps at a time based on their ranks and a specific percent of bad AEs are mutated within some ranges that are proportion to their ranks. This perturbation helps sBCO not to fall into local optimum areas and to get out of the local optimum areas and find new good areas.

Algorithm 1 showed our simple bacteria cooperative optimization algorithm with rank-based perturbation. The basic sBCO algorithm is composed of four processes, initialize $E(t)$, move $E(t)$, sense $E(t)$, and decide $E(t)$. Initialize $E(t)$ sets all initial values of AEs including positions, directions, and the parameters of AEs. Move

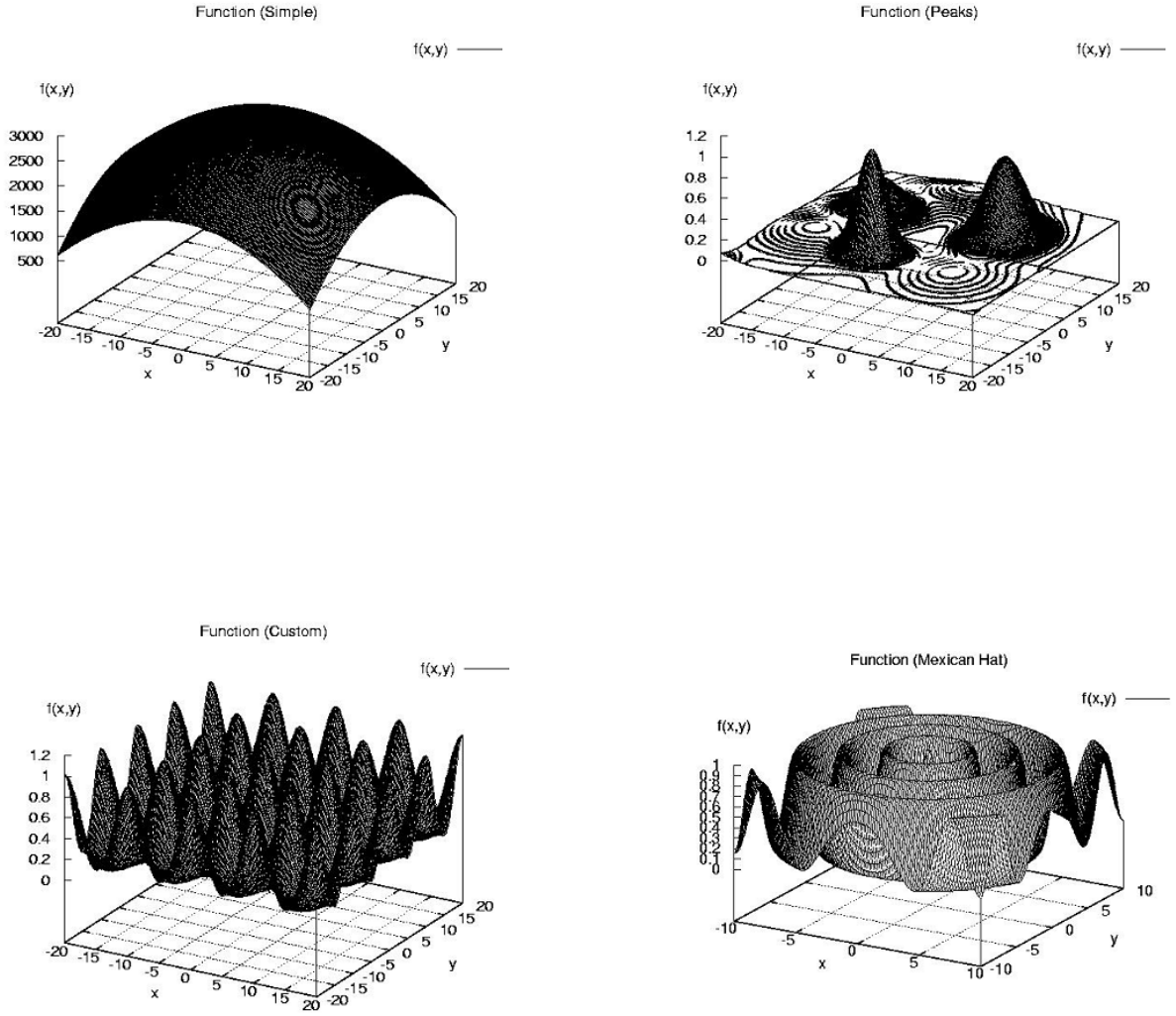

Figure 1. Function optimization problems

$E(t)$ changes the position of AEs corresponding to their direction. In basic sBCO, all AEs move only one step at a time, but AEs in proposed algorithm move v steps at a time assigned according to their ranks. For examples, the best AE

which is rank 1 moves one step and the AE ranked 6 moves 6 steps at a time. For this, rank E(t) process is included in the proposed algorithm. Sense E(t) checks the amount of attractant chemical molecules and calculates the current and previous densities of the attractant chemical molecules according to the first decision rule D1. Decide E(t) is the most important process in that it decides the behaviors of AEs, that is, run or tumble, according to the decision rule D2 and the behavior rules B1 and B2. After decide E(t), mutate E(t) is added to the original algorithm for the mutation of AEs in the proposed algorithm. In this process, a specific percent of bad AEs are mutated within some ranges in proportion to their ranks. That is, the lower the ranks of individuals are, the larger the mutation distances are. The variable speed and mutation of AEs enable that good AEs steadily approach to the global optimum areas while bad AEs find new good areas without staying in local optimum areas for a long time. This effect greatly enhances the performances of sBCO.

IV. EXPERIMENTAL RESULTS

We experimented our proposed algorithm with four function optimization problems, simple function f_1, peaks function f_2, custom function f_3, and Mexican hat function f_4. Their input-output relations are drawn in Figure 1. Function f_1 is relatively simple to optimize because it has only one global optimum area. Function f_2 is somewhat complex than the function f_1 in that it has two relatively large local optimum areas and three small local optimum areas. Since there are many local optimum areas around one global optimum area, function f_3 is quite difficult problem to optimize. Function f_4 called Mexican hat function is another difficult problem because it has too many local optimum areas around the global optimum area.

We set the (B_n, B_m) and (D_n, D_m) to (3, 9) and (3,9) which were known to be closely similar to the parameters of real E. coli. The speed of AEs is set to their ranks and the mutation ranges is proportional to their ranks. The mutation ranges of ith AE are given as: $d_x^i = d_y^i = \left(\mathrm{rand}(r_i) - \frac{r_i}{2}\right) \times \kappa$, where r_i is the rank and κ is a multiplier factor. The number of AEs n and the mutation percent η of bad AEs are also provided for each experiment. All AEs move on the playground repeatedly according to the proposed algorithm and if one AE of them finds the global optimum then its time t is recorded. We experimented for a parameter set with 10 runs and averaged the recorded time t.

Table I showed the averaged values of 10 recorded times with $\kappa = 200, \eta = 0.7$. We also measured the standard deviation of each experiment, but omitted in the table for simplicity. In order to compare our method to the genetic algorithm, we also tested the simple genetic algorithm. In Table I , sGA, sBCO, sBCO_RR, sBCO_VS, sBCO_MUT, sBCO_RBP mean the simple genetic algorithm, simple bacteria optimization, sBCO with rank replacement, sBCO with variable speed, sBCO with mutation, and sBCO with rank-based perturbation, respectively. The sBCO with rank-based perturbation indicates that the two perturbation methods, variable speed and mutation, are applied together. We checked the best result of each experiment with bold underline type. As shown in the table, the rank-based perturbation showed good performances.

fn	n	sGA	sBCO	sBCO_RR	sBCO_VS	sBCO_MUT	sBCO_RBP
f_1	100	10276.8	697.6	176.4	172.1	170.3	**95.8**
	200	9622.1	388.0	158.1	101.8	164.0	**58.4**
	300	4956.0	440.2	146.2	85.4	114.4	**44.1**
f_2	100	308776.0	917.4	238.0	263.4	193.4	**179.1**
	200	352490.8	841.4	183.9	143.4	215.6	**108.6**
	300	97377.0	662.1	158.6	151.0	146.1	**72.4**
f_3	100	14642.0	817.4	200.2	254.5	**171.0**	274.3
	200	13887.4	839.3	197.4	153.6	175.1	**110.4**
	300	15627.2	620.0	174.5	151.5	158.2	**132.6**
f_4	100	40676.5	1155.8	509.5	551.8	734.3	**431.1**
	200	25782.0	639.7	366.0	339.3	**312.9**	363.6
	300	8992.5	681.3	**378.4**	556.9	602.9	494.1

TABLE I. EXPERIMENTAL RESULTS

V. CONCLUSIONS

In this paper, we proposed a simple bacteria cooperative optimization algorithm with rank-based perturbation in order to accelerate the search speed of AEs based on their ranks. As rank-based perturbation, the variable speed and mutation of AEs were employed with some parameters. This perturbation helped the AEs not to fall into local optimum areas and to get out of the local optimum areas and find new good areas for approaching global optimum areas. It was found from experimental results that the search performances of sBCO were improved from the effects of the perturbation.

However, more extensive experiments and more careful analysis about the effects of perturbation and the effects of parameters should be done as a further work.

REFERENCES

[1] M. Dorigo and T. Stutzle, "Ant Colony Optimization, " The MIT Press, 2004.

[2] M. Clerc, "Particle Swarm Optimization, " ISTE Publishing Company, 2006.

[3] L. N. de Castro and J. Timmis, "Artificial Immune Systems: A New Computational Intelligence Approach, " Oxford University Press, 2002.

[4] S. H. Jung and T.-G. Kim, "A Novel Optimization Algorithm Inspired by Bacteria Behavior Patterns," Journal of Korean Institute of Intelligent Systems, vol. 18, pp. 392-400, June 2008.

[5] S. H. Jung, "Simple Bacteria Cooperative Optimization with Rank Replacement," Journal of Korean Institute of Intelligent Systems, vol. 19, pp. 432-436, June 2009.

[6] M. Kim, S. Baek, S. H. Jung, and K.-H. Cho, "Dynamical characteristics of bacteria clustering by self-generated attractants," Computational Biology and Chemistry, vol. 31, pp. 328-334, Oct. 2007.